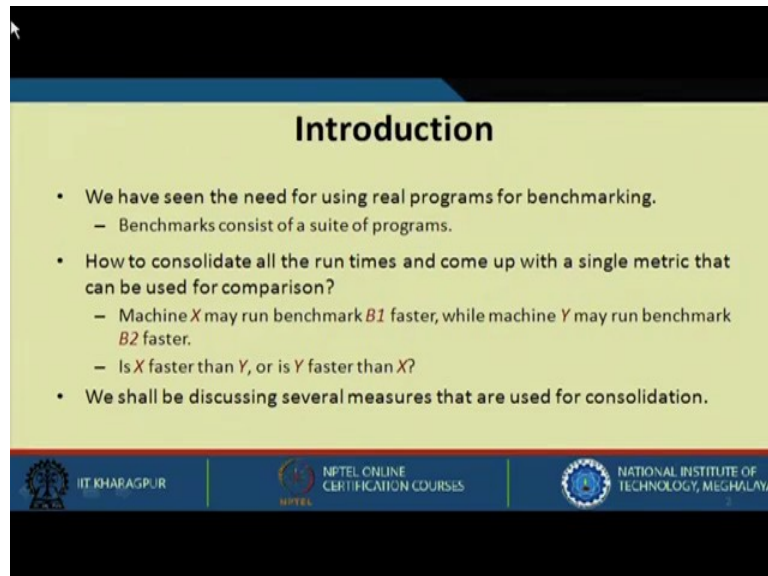


Computer Architecture and Organization
Prof. Kamalika Datta
Department of Computer Science and Engineering
National Institute of Technology, Meghalaya


Lecture – 14
Summarizing Performance Results

(Refer Slide Time: 00:29)



Introduction

- We have seen the need for using real programs for benchmarking.
 - Benchmarks consist of a suite of programs.
- How to consolidate all the run times and come up with a single metric that can be used for comparison?
 - Machine *X* may run benchmark *B1* faster, while machine *Y* may run benchmark *B2* faster.
 - Is *X* faster than *Y*, or is *Y* faster than *X*?
- We shall be discussing several measures that are used for consolidation.



Hello. So, let us come to lecture 14 where we will be summarizing the performance results. What we have seen till now is using real programs for benchmarking like. So, it is very much essential that the benchmark should consist of set of programs, and those programs should be very much relevant.

Now how to consolidate all the run times and come up with a single metric that can be used for comparison? So, we have seen that MIPS is something we are calculating, we are calculating the total execution time. We have so many metrics now. And I can say that a machine A is performing some set of tasks that is taking so and so time. A machine B is also executing the same program and it is taking so on so time. Another set of programs is also been executed by both the processors, and they are giving some time.

Now, it may happen that A is giving better result compared to B for program 1. And B is giving better results for program 2 as compared to program 1. How can you tell that which processor will be better? So, we shall be discussing several measures that are used for this consolidation basically.

(Refer Slide Time: 02:01)

What about Reproducibility?

- Actual run time of a program on a machine depends on so many factors.
 - Degree of multiprogramming, disk usage, compiler optimization, etc.
- Reproducibility of the experiments is very important.
 - Anyone should be able to run the experiment and get the same results.
 - Benchmarks must therefore specify the execution environment very clearly.
- Example:- SPEC benchmarks mention details such as:
 - Extensive description of the computer and the compiler flags.
 - Hardware, software and baseline tuning parameters.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

By the term reproducibility, we mean that we can reproduce the same thing, the same result. Now, I should also say that it is not only that same program will get executed in other machine and we will check the result. I have to also give some more details. What are those some more details? Like what is the execution environment that I am using? How much is the disk utilization? What kind of cache memory I am using? What are the other features along with that program when you are executing?

So, you have to give all those parameters along with the program. Like how much swap space is possible, and so on and so forth. The actual run time of a program on a machine depends on so many factors. One is the degree of multi programming. Like, whether you are using multi programming or not. Disk usage --- how much disk usage is being done, compiler optimization. So, reproducibility of experiments is very important; that means, anyone should be able to run the experiment and get the same result. This can only happen if you actually do the same thing that other computer was doing for executing that program.

So, benchmark must specify the execution environment very clearly. For example, the SPEC benchmarks mentions details such as extensive description of the computer and the compiler flags. So, SPEC benchmark is not only giving you a program, along with that program it is giving you some more details, such that you will be able to reproduce

the same thing, if you execute that particular program. Hardware, software and baseline tuning parameters, like the swap space, like the cache memory, and so on and so forth.

(Refer Slide Time: 04:40)

How to Summarize Performance Results?

- The choice of a good benchmark suite that relates to real applications is essential to measuring performance.
- For a single program, it is very easy to say which computer runs faster.
- However, when there are multiple programs, the comparison may not be so straightforward.

An example

	CPU A (in secs)	CPU B (in secs)	CPU C (in secs)
Program P1	1	10	25
Program P2	500	250	10
Total Time	501	260	35





The slide footer contains logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA.

So, how to summarize performance results? The choice of a good benchmark suite that relates to real applications is essential to measuring performances. The meaning is, the choice of a good benchmark suite is really very important. For a single program it is very easy to say that which computer runs faster. But will it be equally easy, when you have a number of programs running on different machines? And you are getting different results. How can then you say that that this computer performs better than the other? Let us see this example, where programs are run on CPU A, CPU B, and CPU C --- 3 different processors. In seconds, program 1 is taking 1. This is taking 10 and this is taking 25, and these are taking this.

(Refer Slide Time: 05:45)

	CPU A (in secs)	CPU B (in secs)	CPU C (in secs)
Program P1	1	10	25
Program P2	500	250	10
Total Time	501	260	35

- We can make the following statements, which may depict a confusing picture when considered together:
 - A is 10 times faster than B for program P1.
 - B is 2 times faster than A for program P2.
 - A is 25 times faster than C for program P1.
 - C is 50 times faster than A for program P2.
 - B is 2.5 times faster than C for program P1.
 - C is 25 times faster than B for program P2.



So, total time taken is this. Now let us see. We can make the following statements, which may depict a confusing picture, when considering together. If you consider it separately, for program 1, A is the fastest; it takes 1 second. And for program 2, C is the fastest, we can say. Now, let us say this with respect to A; A is 10 times faster than B. Very true, A takes 1 second, B takes 10 seconds. Similarly, B is 2 times faster than A for program P2. This is for program P1. Now B, which is taking 250 second and program P2 for processor A, it is taking 500.

So, this is taking twice the time. So, we can say B is 2 times faster than A for program P2. Again, A is 25 times faster than C for program P2. And C is 50 times faster than A for program P2. Because P2, this takes 500 this is taking only 10. Similarly, you can find out B is 2.5 times faster than C for program P1. C is 25 times faster than B for program P2. Now how can I say which processor is the best? It is more confusing, rather than getting some inference.

So, how to summarize this performance result? Let us see how we can do this. Choice of a good benchmark suite that relates to real applications is essential to measuring performance. So, for a single program it is very easy to say, but; however, when there are multiple programs, the comparison may not be so straightforward. So, we need to come up with some solution.

(Refer Slide Time: 08:02)

(a) Total Execution Time

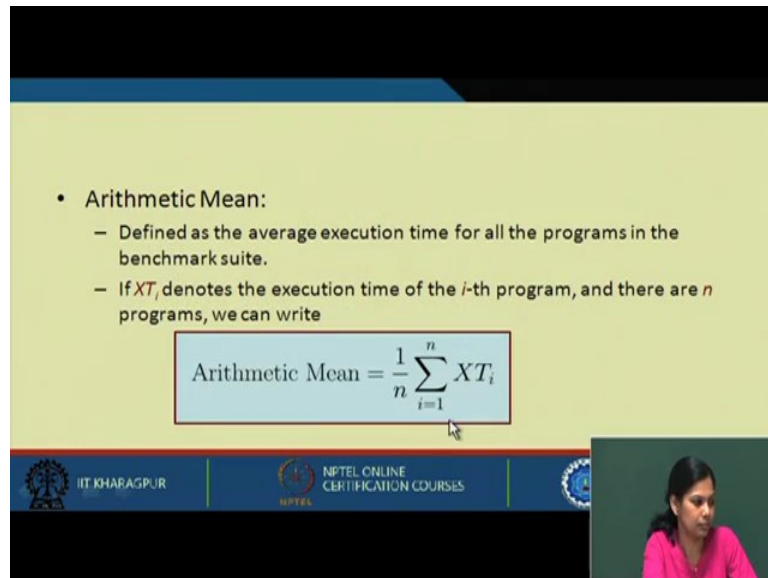
- The simplest approach to summarize the relative performances is to look at the total execution times of the two programs.
 - CPU A : 501, CPU B: 260, CPU C: 35.
- Based on this measure, we can make the following comments:
 - B is $501 / 260 = 1.93$ times faster than A for the two programs.
 - C is $501 / 35 = 14.31$ times faster than A for the two programs.
 - C is $260 / 35 = 7.43$ times faster than B for the two programs.
- If the actual workload consists of running P1 and P2 unequal number of times, this measure will not give the correct result.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, let us see the total execution time first. What is the total execution time? For CPU A, the total execution time is 501. For CPU B, the total execution time is 260. And for CPU C, it is 35. Based on this measure what we can see or what we can comment is, B is 1.93 faster than A for the 2 programs.

Now, we are not taking into consideration individual programs. Now we are saying something in terms of both the programs. C is 14.31 times faster than A for 2 programs. And C is 2 point 7.43 times faster than B for 2 programs. If the actual workload consists of running P1 and P2 unequal number of times, again this is something else. First we have given 2 programs and we are saying this is the execution times. And now we are saying that these programs how many times they will be running. So, the frequency of running those programs is now coming into picture.

(Refer Slide Time: 09:27)



• Arithmetic Mean:

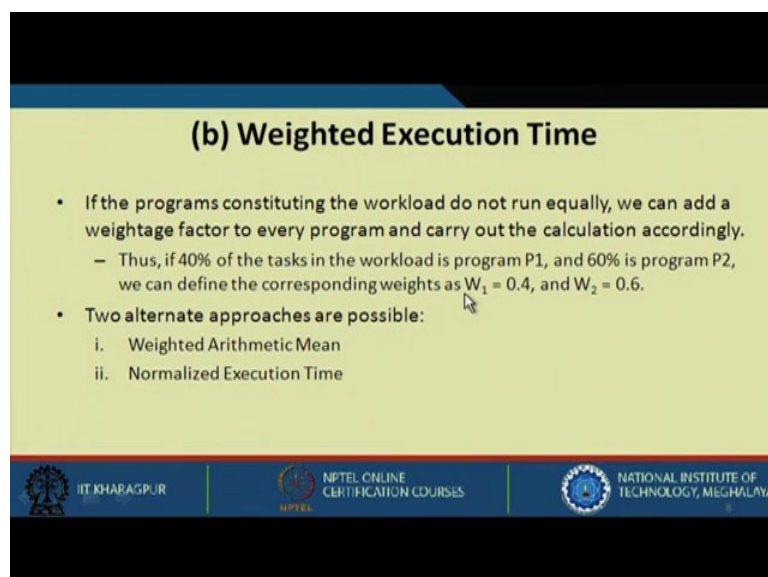
- Defined as the average execution time for all the programs in the benchmark suite.
- If XT_i denotes the execution time of the i -th program, and there are n programs, we can write

$$\text{Arithmetic Mean} = \frac{1}{n} \sum_{i=1}^n XT_i$$

The slide includes logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and NPTEL. A small video inset shows a woman in a pink shirt.

So, if the actual workload consists of running P1 and P2 unequal number of times, this measure will not give the correct result either. So, now let us come up with something called arithmetic mean. So, this is defined as the average execution time, for all the programs in the benchmark suite. By average execution time what we mean is that, we are averaging the execution time summation of all divided by n . This is your arithmetic mean where XT_i denotes the execution time of i -th program. And there are n programs.

(Refer Slide Time: 10:02)



(b) Weighted Execution Time

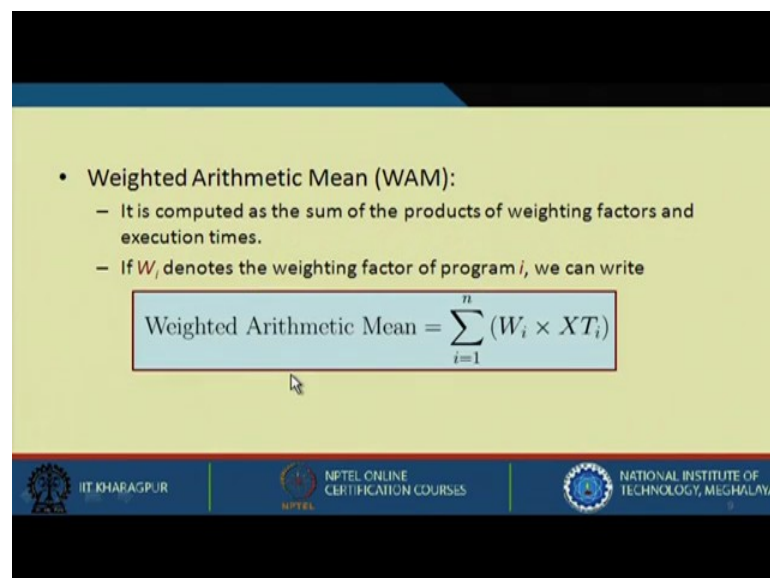
- If the programs constituting the workload do not run equally, we can add a weightage factor to every program and carry out the calculation accordingly.
 - Thus, if 40% of the tasks in the workload is program P1, and 60% is program P2, we can define the corresponding weights as $W_1 = 0.4$, and $W_2 = 0.6$.
- Two alternate approaches are possible:
 - i. Weighted Arithmetic Mean
 - ii. Normalized Execution Time

The slide includes logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA.

Similarly, what is weighted execution time? If the program constituting the workload do not run equally, as I said program A can run more times than program B. For some processor; program B can run more times. So, based on this how you can calculate.

So, for this if 40% of the task in workload is program P1, and 60% is program P2. We can define some weights associated with these programs. So, program P1 is having weight $W_1 = 0.4$. And $W_2 = 0.6$. Because P1 it is 40% of the tasks of the workload, and 60% for P2. Now we will see two alternative methods. One is weighted arithmetic means. Another is normalized execution time.

(Refer Slide Time: 11:21)



• **Weighted Arithmetic Mean (WAM):**

- It is computed as the sum of the products of weighting factors and execution times.
- If W_i denotes the weighting factor of program i , we can write

$$\text{Weighted Arithmetic Mean} = \sum_{i=1}^n (W_i \times XT_i)$$

The slide also features logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA at the bottom.


Now, arithmetic mean we have already seen. Let us see what is weighted arithmetic mean (WAM). It is computed as a sum of products of weighting factors and the execution times. So, we are not only considering only the execution time, we are taking into consideration the execution time and the weights associated with that particular execution time where W_i denotes the weighting factor of program i . This is the weighted arithmetic mean where we multiply W_i by execution time of i . And we take the sum.

(Refer Slide Time: 11:56)

Example 1

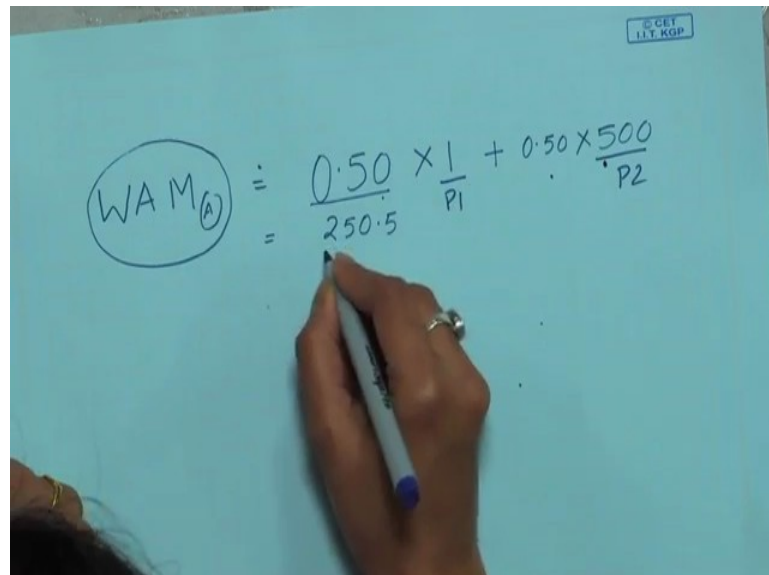
	CPU A (in secs)	CPU B (in secs)	CPU C (in secs)
Program P1	1	10	25
Program P2	500	250	10
Total Time	501	260	35

- If $W_1 = 0.50$ and $W_2 = 0.50$, we get $WAM_A = 250.5$, $WAM_B = 130$, $WAM_C = 17.5$
- If $W_1 = 0.90$ and $W_2 = 0.10$, we get $WAM_A = 50.9$, $WAM_B = 34$, $WAM_C = 23.5$
- If $W_1 = 0.10$ and $W_2 = 0.90$, we get $WAM_A = 450.1$, $WAM_B = 226$, $WAM_C = 11.5$



Now, let us see these 2 programs again. So, weight for first one is 0.50 and weight of second one is 0.50. So, we get weighted arithmetic mean of A as 250.5. How we are getting that? Let us see this. So, here you can see that 0.50 is multiplied by 1 for program P1 for CPU A.

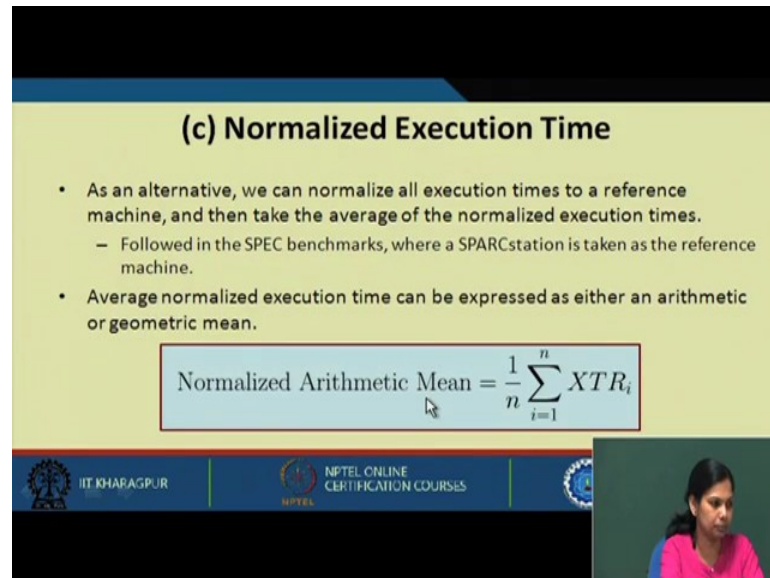
(Refer Slide Time: 12:29)


$$\begin{aligned} WAM(A) &= \frac{0.50}{1} \times \frac{1}{P1} + 0.50 \times \frac{500}{P2} \\ &= 250.5 \end{aligned}$$

So, this is CPU weighted arithmetic mean for A. We are multiplying this weight with the time that is 1. And similarly, we are multiplying 0.50 by 500 which is for program P2. And we are getting weighted arithmetic mean as 250.5. Weighted arithmetic mean for B

is this, and for C is this. Depending on these values, we have calculated and depending on these weights. Similarly, if you change the weight of these, the weighted arithmetic mean is different. Again if you change the weighted arithmetic mean is different. So, we here also we show you, how we can calculate the weighted arithmetic mean for several CPUs given the execution time for some programs.

(Refer Slide Time: 13:51)



(c) Normalized Execution Time

- As an alternative, we can normalize all execution times to a reference machine, and then take the average of the normalized execution times.
 - Followed in the SPEC benchmarks, where a SPARCstation is taken as the reference machine.
- Average normalized execution time can be expressed as either an arithmetic or geometric mean.

$$\text{Normalized Arithmetic Mean} = \frac{1}{n} \sum_{i=1}^n XTR_i$$

The slide also features logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and NPTEL, along with a small video inset of a presenter.

Now, coming to normalized execution time. As an alternative we can normalize all execution times to a reference machine. And then take the average of the normalized execution times. So, now we are saying that there will be a reference machine, and with respect to that reference machine we will be calculating this. So, we are normalizing it with respect to a reference machine. Followed in SPEC benchmarks where a SPARC station is taken as the reference machine. So, what will be the average normalized execution time? So, this can be expressed as execution time, what is XTR ? Execution time, with the reference XTR_i and summation of all of these.

So, one machine can be taken as a reference and can be calculated based on that. We will be seeing that next. That is called normalized arithmetic mean. Which is execution time of the programs with respect to reference machine summation of that divided by n .

(Refer Slide Time: 15:11)

Normalized Geometric Mean = $\sqrt[n]{\prod_{i=1}^n XTR_i}$

- Here, XTR_i denotes the execution time for the i -th program, normalized to the reference machine.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

So, here XTR_i denotes the execution time for i -th program normalized to the reference machine. This is normalized geometric mean. Earlier it was normalized arithmetic mean. Now we are going to normalize geometric mean.

(Refer Slide Time: 15:38)

Example 2

	CPU A (in secs)	CPU B (in secs)	CPU C (in secs)
Program P1	1	10	25
Program P2	500	250	10
Total Time	501	260	35

	Normalized to A			Normalized to B			Normalized to C		
	A	B	C	A	B	C	A	B	C
Program P1	1.0	10.0	25.0	0.1	1.0	2.5	0.04	0.4	1.0
Program P2	1.0	0.5	0.02	2.0	1.0	0.04	50.0	25.0	1.0
Arithmetic mean	1.0	5.25	12.51	1.05	1.0	1.27	25.02	12.7	1.0
Geometric mean	1.0	2.24	0.71	0.45	1.0	0.32	1.41	3.16	1.0

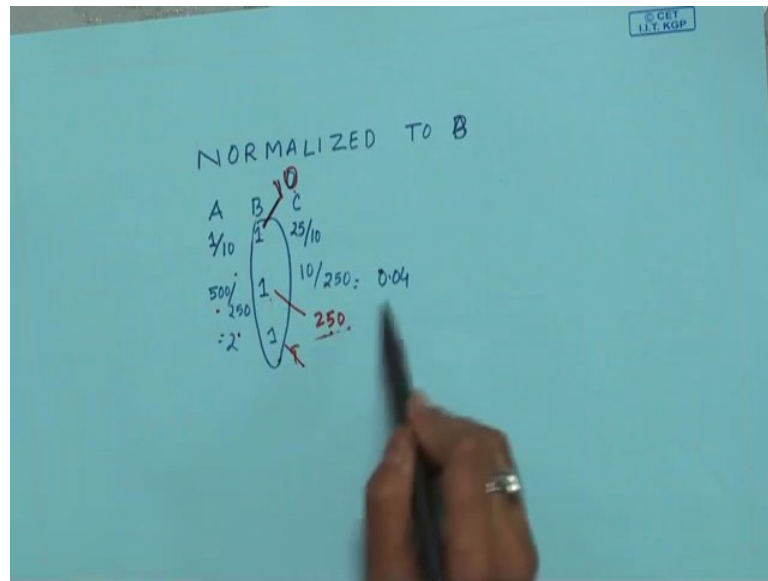
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Now, let us see how do we compute this. So, program 1 has the same parameter that we were using. Now see what we are doing. These values we are calculating by with respect to normalized to A; that means, when we are saying normalized to A we are making 1 here, these 2 will become 1. And with reference to that what are these values. So, now

see this is 1. 10 divided by 1.25 divided by 1. Similarly, this one is, how do you get 1 here? You will get 1 here by dividing by 500.

So, this has become 1 you divide by 500 it will become 0.02. So, all these values are normalized with respect to A.

(Refer Slide Time: 16:49)



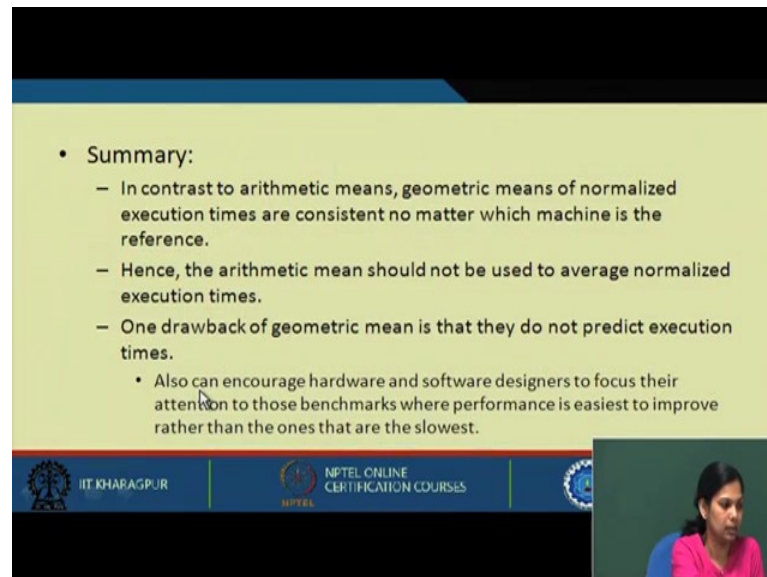
Let us see, how do we normalize with respect to B similarly. When we say we normalized with respect to B, then we have to make all these as 1, 1, 1. Initially it was 10 this was 250, and 0. So, what we are doing? We are making others reference normalized to B. So, what we are doing; this is 1 divided by 10, and this will become 25 divided by 10, and here it was 250. So, you have to divide it by 250.

So, we are dividing 500 divided by 250 to make it 1. So, we will get here 1. And 10 divided by 250. So, just see this what we are getting. We are normalized this with respect to 1, we are making this as 1. And now these are the 2 values we are getting. And similarly we have normalized this 2, see these 2 have become 1. And we are getting this value with respect to the difference machine. Now once we are done with this we calculate the arithmetic mean and we calculate the geometric mean.

Now, say arithmetic means still we cannot say which one is better. So, in this case this is one. In this case this is one. And in this case this one. So, it is very difficult to come up with a conclusion based on this, but you see the geometric mean seems to be consistent.

C is taking the least for here, also here. So, the geometric mean is considered as one of the metric that can be used for evaluation. Although arithmetic mean cannot be used properly.

(Refer Slide Time: 19:11)



- Summary:
 - In contrast to arithmetic means, geometric means of normalized execution times are consistent no matter which machine is the reference.
 - Hence, the arithmetic mean should not be used to average normalized execution times.
 - One drawback of geometric mean is that they do not predict execution times.
 - Also can encourage hardware and software designers to focus their attention to those benchmarks where performance is easiest to improve rather than the ones that are the slowest.

So, in summary what we can say that, in contrast to arithmetic mean or geometric mean normalized to the execution time are consistent, no matter, which machine is the reference one. So, what we have seen, we have made A as a reference machine then also see C is better. We have made B as a reference machine in that case also C is giving better. And when you are making C as the reference machine then also it is giving better. So, the result seems to be consistent, even if different programs are executed different times.

Hence arithmetic mean should not be used to average normalized execution times, but there is one drawback of geometric means that they do not predict the execution time. You cannot give a prediction of the execution time, but it is consistent because it is giving that value where C seems to be better. And it also encouraged hardware and software designers to focus their attention to those benchmarks where performance is easiest to improve rather than the ones that are slowest. So, we generally improve where the requirement is high; that means, certain instruction are executed more we try to improve that part more. Certain instruction and executed less, but that does not mean we will leave that part also. That part also we should take into consideration. But it generally

encourages the hardware and software designer to focus attention at those benchmarks where performance is easiest to improve. That is the thing.

So, we came to the end of this lecture. So, in the last three lectures what we have tried to show is, what is performance. What performance metrics can be used?

Thank you.